CROSS-JEM: Accurate and Efficient Cross-encoders for Short-text Ranking Tasks

Bhawna Paliwal [1*†]   **Deepak Saini** [2*†]   **Mudit Dhawan** [3‡]

Siddarth Asokan [1†]   **Nagarajan Natarajan** [1]   **Surbhi Aggarwal** [4]

Pankaj Malhotra [4]   **Jian Jiao** [2]   **Manik Varma** [1]

[1]Microsoft Research, India [2]Microsoft, USA [3]Carnegie Mellon University [4]Microsoft, India

{bhawna, desaini}@microsoft.com, mdhawan@cs.cmu.edu,

{siddarth.asokan, Nagarajan.Natarajan, Surbhi.Aggarwal}@microsoft.com,

{pamalhotra, Jian.Jiao, manik}@microsoft.com

Abstract

Ranking a set of items based on their relevance to a given query is a core problem in search and recommendation. Transformer-based ranking models are the state-of-the-art approaches for such tasks, but they score each query-item independently, ignoring the joint context of other relevant items. This leads to sub-optimal ranking accuracy and high computational costs. We address this by proposing Cross-encoders with Joint Efficient Modeling (CROSS-JEM), a novel ranking approach that enables transformer-based models to jointly score multiple items for a query, maximizing parameter utilization. CROSS-JEM leverages (a) redundancies and token overlaps to jointly score multiple items, that are typically short-text phrases arising in search and recommendations, and (b) a novel training objective that models ranking probabilities. CROSS-JEM achieves state-of-the-art accuracy and over 4x lower ranking latency over standard cross-encoders. Our contributions are threefold: (i) we highlight the gap between the ranking application's need for scoring thousands of items per query and the limited capabilities of current cross-encoders; (ii) we introduce CROSS-JEM for joint efficient scoring of multiple items per query; and (iii) we demonstrate state-of-the-art accuracy on standard public datasets and a proprietary dataset. CROSS-JEM opens up new directions for designing tailored early-attention-based ranking models that incorporate strict production constraints such as item multiplicity and latency.

1 Introduction

We consider the problem of ranking that arises in search and recommendation pipelines, wherein the goal is to rank a set of items based on their relevance to a given query. Our work is in the context of two-stage *retrieve-then-rank* pipelines in modern search engines [1–6] as depicted in Figure 1. Given a *query*, i.e., a search phrase such as *"patagonia japan"*, the retrieval stage pares the *items*, i.e., keywords such as *"clothing store japan"*, *"patagonia homes shimoda"* that are bid against for displaying ads, from billions to a few hundreds [7, 8] of potential interest. These items are subsequently provided as the input to the ranking stage. In this work, we focus on the ranking model, given a black-box retriever. We consider short-text items (as in the above example), which appear in a myriad of recommendation systems applications such as product recommendation, query to advertiser bid phrase recommendation, and Wikipedia category tagging [9–11]. In designing the

---

[*]Denotes equal contribution

[†]Corresponding authors

[‡]Work done while at Microsoft Research, India
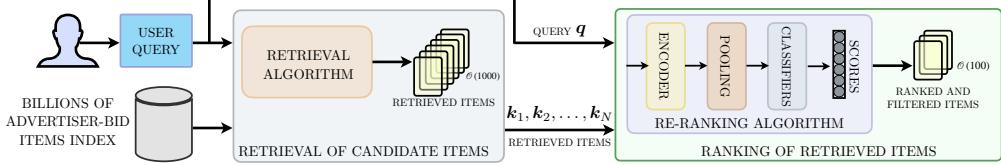
Figure 1: The two-stage large-scale search and recommendation pipeline comprises: (i) candidate selection from billions of items; (ii) re-ranking of retrieved items. CROSS-JEM works at stage (ii), to output the score for all N item in a single pass.

Table 1: A comparison of CROSS-JEM's listwise modeling and a pointwise ranking model [2]: Relatively more *generic* (but relevant) items are ranked higher in baseline predictions, owing to biases such as high frequency in the training data. Our listwise model evaluates all the shortlisted items in a single forward pass, and ranks more specific (and relevant) items higher.

Query	Top-5 ranked items in the Proposed Approach (CROSS-JEM)	Top-5 ranked items in the baseline Cross Encoder
different foods of oaxaca mexico	'the foods of oaxaca', 'oaxacan cuisine', 'exploring oaxacan food', 'authentic recipes from oaxaca, mexico', '6 things you'll love about oaxaca'	'culinary tales: the kinds of food mexicans eat', 'mexican christmas foods', 'mexican cuisine', 'culture: food and eating customes in mexico', 'popular food in mexico',
what is the bovine growth hormone	'recombinant bovine growth hormone', 'what is rbgh?', 'rbgh', 'bovine growth hormone and milk : what you need to know', 'what is rbst?'	'growth hormone', 'human growth hormone', 'alternative names for growth hormone', 'human growth hormone and insulin are friends', 'growth hormone (somatotropin)'

ranking model, two key axes are the model architecture, and the choice of the loss function, while the key performance metrics for such systems are accuracy and inference latency.

Traditionally, *encoders* that employ stacked attention layers to encode a query-item pair, followed by a *classifier* to predict ranking scores are widely adopted for ranking [2, 12, 5]. Recently, sequence-to-sequence models with encoder-decoder and decoder-only architectures have also been proposed for ranking. These models provide a ranking score based on particular vocabulary-token logits [13–15]. However, these approaches model ranking as a *pointwise* task, providing ranking scores for a given query and item pair. But ranking is inherently a list-based task that requires scoring query-item pairs relative to one another, and not in isolation. In particular, the other items in the list to be ranked for a given query provide crucial context for scoring. Pointwise models neglect the list context, produce independent scores that may not reflect the optimal ranking order, and are difficult to calibrate across items for sorting to provide final rankings [16]. Table 1 illustrates this by juxtaposing the top-5 ranked items obtained using our proposed approach (listwise modeling) and a baseline pointwise ranking model [2]. We observe that relatively more *generic* items *e.g.,* "*mexican cuisine*", although relevant to the query "*different foods of oaxaca mexico,*" are ranked higher in baseline predictions, owing to (a) their frequency in training data, (b) token-level matching and other biases which are difficult to mitigate in pointwise modeling. On the other hand, our proposed listwise approach evaluates all the items to be ranked holistically, and subsequently ranks more specific (not just relevant) items higher.

Furthermore, pointwise transformer based models [2, 14] are computationally expensive and impractical for real-time ranking systems that need to handle large-scale traffic requiring low latency and high throughput. Therefore, many industrial systems resort to using simpler sparse neural networks [17] or late-interaction models [18, 19] for online ranking, sacrificing accuracy for latency.

Research along listwise loss functions [20, 14, 21] aim to improve ranking accuracy by optimizing training objective for the whole list of items, not just query-item pairs. Yet, architecturally, they score items independently, ignoring inter-item dependencies and query context. Recent works also use pre-trained LLMs for listwise ranking [22–24, 15]. However, these models have a huge parameter count (a few billions), limiting their scalability and efficiency. **In this work, we bridge this gap by proposing a ranking model that works at the list level, explicitly models inter-item interactions, and achieves superior latency-accuracy tradeoff, making it deployable in real-time scenarios.**

The Proposed Approach: We propose an end-to-end joint ranking approach that models the listwise ranking of the query and given items, capturing both the query-item and the item-item interactions and satisfying the strict latency requirements of industry-scale recommendation systems. Our approach, entitled **CROSS-JEM** (**CROSS**-Encoder with **J**oint **E**fficient **M**odeling), leverages the list structure of the input in both the encoder and the classifier components, as well as in the training objective.

While any standard loss function (e.g., binary cross entropy) could be used to train a CROSS-JEM model, in this work, we propose a novel variant of the listwise loss functions [21, 25], Ranking Probability Loss (RPL); which can be interpreted as a divergence between the predicted probabilities and the estimated ranking probabilities (as a function of model logits and ground truth rankings) of items to be ranked. The proposed listwise loss works much better in conjunction with our joint modeling than standard pointwise or listwise losses [21, 25, 26]. To the best of our knowledge, we are the first to propose a joint ranking approach that can effectively model listwise ranking in both the model architecture and the training objective with real-time latency constraints.

In summary, our key contribution is the introduction of a novel **joint ranking approach CROSS-JEM**, that scores multiple items per query in a single pass, exploiting token interaction across items for better accuracy and token redundancies for better efficiency. We demonstrate the effectiveness of CROSS-JEM on two public benchmark datasets for short-text re-ranking, wherein it outperforms the best-performing baselines by at-least 3% in terms of MRR. When applied to large-scale search-based recommendation, CROSS-JEM demonstrated a 13% higher accuracy than state-of-the-art models, while being over $6\times$ faster than standard cross-encoders [2]. We also deploy CROSS-JEM for real-time ranking on live traffic, where it reduces the quick-back-rate by 1.8%, indicating improved relevance of ads to users. Our work presents a general, scalable framework for joint ranking of multiple items across various domains and short-text tasks, accounting for ranking under task-specific constraints such as item multiplicity and latency.

2 Background and Related Work

Ranking Architecture: State-of-the-art transformer based models with stacked attention layers are typically encoder based, such as monoBERT and Birch [2, 27]. These models encode query-passage pairs with a bi-directional attention encoder and use a classifier to obtain a ranking score. Alternatively, sequence-to-sequence models, such as monoT5 [13] leverage the pre-training knowledge of generative models for ranking. These models generate a ranking score from a specific vocabulary token in the decoder output. Another line of work explores decoder-only models, such as llama- and GPT-based models and rely on their extensive pre-training knowledge and parameter count for ranking [15, 28]. Despite their advantages, decoder-only models fine-tuned on large ranking corpora [15] do not outperform the fine-tuned encoder-based and sequence-to-sequence models on short-text ranking tasks (cf. Section 5). CROSS-JEM is the first joint-ranking approach that effectively incorporates listwise ranking into the model architecture and training objective while maintaining latency constraints.

Joint (Listwise) Ranking: Ranking inherently involves comparing a list of items; thus, recent works employ listwise loss functions in encoder or encoder-decoder models to enhance ranking accuracy. Gao et al. [29] devised a listwise multi-class cross-entropy loss to optimize ranking probabilities in encoder models. Zhuang et al. [14] introduced a ranking-specific listwise cross-entropy loss to improve performance of Seq2Seq models for ranking. They also demonstrated that an expanded form of cross-entropy loss (poly-1) achieved superior performance across various ranking metrics. Although listwise losses improved ranking accuracy, the model architectures of these rankers remain pointwise and do not fully capture the ranking task. CROSS-JEM ranks multiple items per query in one pass, capturing item-item interactions in the ranking list to improve accuracy. It is optimized with Ranking Probability Loss (RPL), a novel variant of the listwise ranking loss (as considered in ListNet [21]) aligned with the CROSS-JEM architecture, and estimates the ranking probabilities using target relevance labels and predicted model logits.

LLMs for Ranking: LLMs have emerged as a powerful tool for ranking tasks; they can leverage pre-trained knowledge, large parameter count, and effective prompting to achieve superior performance. Existing works have adopted two main approaches to exploit LLMs for ranking: (a) Using LLMs directly as re-rankers by designing novel prompting schemes [30, 22, 23] and sorting strategies [24, 31]; the high-level idea is to encode the ranking items and the query into a single input and generate a ranked list as output, using a sliding window technique to handle long inputs; (b) Using LLMs for learning more accurate smaller ranking models [22, 32, 15, 33] via standard distillation or via augmenting the training data with synthetic samples generated by the LLM. These works have demonstrated that LLMs can outperform small-scale supervised methods [2, 13, 24] on various ranking benchmarks. Deploying LLMs at scale for real-time serving scenarios is still challenging due to their high computational costs and memory requirements. Using LLMs to enhance smaller ranking models (as in (b)), is practical and scalable, and can be applied to CROSS-JEM as well.

Efficiency and Scaling: Another closely related area is the focus on improving the efficiency and scaling of transformer based rankers (and retrievers) using light-weight architectures. Approaches employing early-attention (such as monoBERT [2]) have shown high ranking accuracies, but cannot support the low-latency requirements of online raking applications in industry-scale recommendation systems. This can be attributed to the requirement of making multiple calls to the expensive transformer-based encoder to rank a list of items per query, which is infeasible in a few milliseconds. Therefore, online production systems use a variation of sparse neural networks, namely MEB [17], for ranking thousands of items in real-time. Late-interaction models such as ColBERT [19], Baleen [34] and TwinBERT [18] are also used for online ranking due to their computational efficiency. These models reduce the computational costs by applying a late-interaction layer over query-item embeddings. This comes at the price of accuracy, owing to the lack of interactions between query and item tokens. They also incur high storage overheads in online settings, as they need to store and retrieve token-level embeddings. Another line of work that focuses on efficient retrieval and ranking is based on dual-encoder architectures, such as ANCE [35], DPR [36], and INSTRUCTOR [37]. These methods use contrastive-style training to learn a query and item encoder and metrics such as cosine similarity to rank query-item pairs. They can be made highly efficient via a nearest neighbor search [38, 39] over pre-computed embeddings, but lose out on accuracy compared to cross-encoders [2, 40].

3 CROSS-JEM: CROSS-encoders with Joint Efficient Modeling

We now present CROSS-JEM, our proposed approach to accurate and low-latency ranking via efficient scoring of multiple items. Before we describe CROSS-JEM in detail, we develop the notation used in the subsequent sections of this manuscript.

Notation: We denote queries by q and the corresponding set of N candidate items retrieved for q by $\mathbb{K}_q = \{k_1, k_2, ..., k_N\}$. We denote the dataset of queries and items used for training by $\mathbb{Q}_{tr}$ and $\mathbb{I}_{tr}$ respectively, and that of the test datasets by $\mathbb{Q}_{te}$ and $\mathbb{I}_{te}$. We drop the subscripts when the meaning is clear from the context. The ground-truth scores are given by $y_i \in \mathbb{R}^N$, $[y_i]_j = y_{ij}$ denoting the score for the item k_j from set $\mathbb{K}_{q_i}$, associated with the query q_i. The queries q and items k_j are tokenized via $\mathcal{T}(\cdot)$ to obtain d-dimensional representations (tokens), given by $\mathbb{T}_q = \{\mathbf{q}^1, \mathbf{q}^2, \ldots, \mathbf{q}^{L_q}\}$, and $\mathbb{T}_{k_j} = \{\mathbf{k}_j^1, \mathbf{k}_j^2, \ldots, \mathbf{k}_j^{L_{k_j}}\}$, respectively, where $\mathbf{q}^\ell, \mathbf{k}_j^\ell \in \mathbb{R}^d$.

Research Problem: We seek to learn a ranking model that, given a query q, assigns scores $\mathbb{S}_q = \{s_1, s_2, ..., s_N\}$ for all items in $\mathbb{K}_q$, such that the ranking induced by the scores is accurate.

3.1 The CROSS-JEM Architecture

CROSS-JEM comprises an encoder to obtain representations of a given query q and all its candidate items $\mathbb{K}_q$. The representations are pooled and passed to a classification head, that outputs a score corresponding to each item $k_i \in \mathbb{K}_q$ associated with the query.

CROSS-JEM's architecture is primarily inspired by the observation that the candidate items in the set $\mathbb{K}_q$, for a given query q, has significant token overlap amongst themselves. A more in-depth exploration of this phenomenon, in the context of efficient-scoring, is provided in Appendix G. Given a query q and its candidate items $\mathbb{K}_q$, the core idea is we form the union of tokens $\mathbb{T}_{U_q}$ of all items in $\mathbb{K}_q$. Subsequently, the representations of query and set $\mathbb{T}_{U_q}$ can be obtained in a *single pass* of the encoder, and scored via a *single pass* over the classifiers.

While at first glance, it might appear that using $\mathbb{T}_{U_q}$ (with a potential loss of item token ordering) could adversely affect performance, we observed in preliminary experimentation that this is not the case when scoring short-text items. In particular, we compared the performance of two cross-encoder models on search engine logs, one with items as-is, and another comprising items with alphabetically sorted tokens. Both the mean average precision (MAP) and accuracy of the latter model was found to be within 1% of the score obtained when the sequence information is retained. Additional discussions on sequence information are provided in Appendix G. We now describe the CROSS-JEM encoder and classifier in detail.

Encoder: CROSS-JEM employs a trainable encoder $\mathcal{E}_\theta$, which takes as input a sequence of tokens $\mathbf{T} = [\mathbf{t}^1, \mathbf{t}^2, \ldots, \mathbf{t}^n]$ (of length n), and generates as output another sequence of d-dimensional contextual embeddings $\mathcal{E}_\theta(\mathbf{T}) = \mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n]$. These embeddings provide context-
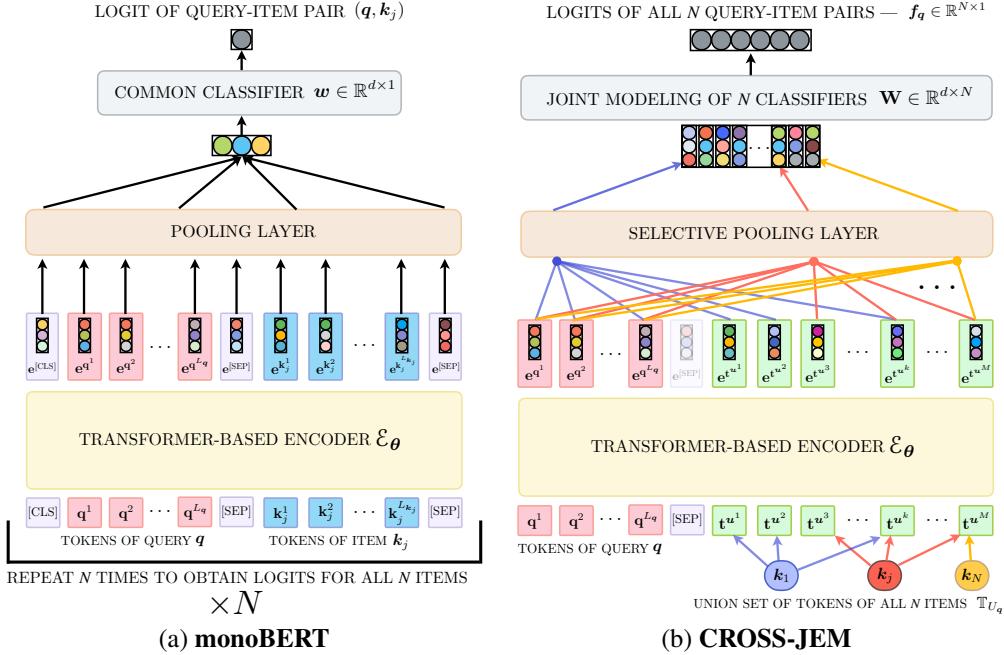
Figure 2: (🌈 Best viewed in color) Computing the relevance scores for query $\boldsymbol{q}$ and retrieved set of N items $\{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_N\}$ using (a) monoBERT and (b) CROSS-JEM: (a) monoBERT (standard cross-encoders) infers the scores for each $(\boldsymbol{q}, \boldsymbol{k}_j)$ pair with query and item tokens individually (pointwise). The inference is repeated N times for N items. (b) CROSS-JEM jointly inputs query tokens and the set of union of tokens from all items to be ranked (listwise). The logits for an item $\boldsymbol{k}_j$ can be computed by selecting the contextual embeddings corresponding to its tokens. The selected embeddings are pooled, and input to the linear classifier jointly (*i.e.*, $\mathbf{W} = [\boldsymbol{w}, \boldsymbol{w}, \ldots, \boldsymbol{w}] \in \mathbb{R}^{d \times N}$), thus obtaining the N scores in a single encoder and classifier pass.

dependent representations of the input tokens, and can be used for downstream tasks such as classification, generation, and retrieval. Given the tokenization of the query $(\mathbb{T}_{\boldsymbol{q}})$, and that of an item $(\mathbb{T}_{\boldsymbol{k}_j})$, the contextual embeddings of the tokens $[\mathbf{t}^{[\text{CLS}]}, \mathbf{q}^1, \mathbf{q}^2, \ldots, \mathbf{q}^{L_{\boldsymbol{q}}}, \mathbf{t}^{[\text{SEP}]}, \mathbf{k}_j^1, \mathbf{k}_j^2, \ldots, \mathbf{k}_j^{L_{\boldsymbol{k}_j}}]$ in baseline variants is given by $[\mathbf{e}^{[\text{CLS}]}, \mathbf{e}^{\mathbf{q}^1}, \mathbf{e}^{\mathbf{q}^2}, \ldots, \mathbf{e}^{\mathbf{q}^{L_{\boldsymbol{q}}}}, \mathbf{e}^{[\text{SEP}]}, \mathbf{e}^{\mathbf{k}_j^1}, \mathbf{e}^{\mathbf{k}_j^2}, \ldots, \mathbf{e}^{\mathbf{k}_j^{L_{\boldsymbol{k}_j}}}]$, where $\mathbf{e}^{[\text{CLS}]}$ and $\mathbf{e}^{[\text{SEP}]}$ denote the embeddings of the $\mathbf{t}^{[\text{CLS}]}$ and $\mathbf{t}^{[\text{SEP}]}$ tokens, respectively. These contextual embeddings are pooled to obtain a single d-dimensional embedding for each pair $(\boldsymbol{q}, \boldsymbol{k}_j)$ by means of a sum or mean pooling layer, or by taking $\mathbf{e}^{[\text{CLS}]}$. This process is computationally expensive due to the need for N forward passes of the encoder to compute the scores for each item in $\mathbb{K}_{\boldsymbol{q}}$.

In CROSS-JEM, we leverage the short-text nature of the items, item-item interactions, and redundancy of tokens amongst items in $\mathbb{K}_{\boldsymbol{q}}$. This is done by computing the contextual embeddings for all *distinct tokens in the retrieved item set* $\mathbb{K}_{\boldsymbol{q}}$ in one pass over the sequence of the query tokens $\mathbb{T}_{\boldsymbol{q}}$ combined with *item token union set* $\mathbb{T}_{U_{\boldsymbol{q}}} = \left\{ \mathbf{t}^{\boldsymbol{u}^1}, \mathbf{t}^{\boldsymbol{u}^2}, \ldots, \mathbf{t}^{\boldsymbol{u}^M} \right\}$, where M is the total number of tokens in the union set. The contextual embeddings of all input tokens in CROSS-JEM are computed as

$$\mathbf{E} = \mathcal{E}_{\boldsymbol{\theta}} \left(\left[\mathbf{q}^1, \ldots, \mathbf{q}^{L_{\boldsymbol{q}}}, \mathbf{t}^{[\text{SEP}]}, \mathbf{t}^{\boldsymbol{u}^1}, \ldots, \mathbf{t}^{\boldsymbol{u}^M} \right] \right) = \left[\mathbf{e}^{[\text{CLS}]}, \mathbf{e}^{\mathbf{q}^1}, \ldots, \mathbf{e}^{\mathbf{q}^{L_{\boldsymbol{q}}}}, \mathbf{e}^{[\text{SEP}]}, \mathbf{e}^{\mathbf{t}^{\boldsymbol{u}^1}}, \ldots, \mathbf{e}^{\mathbf{t}^{\boldsymbol{u}^M}} \right].$$

Since the number of tokens in the item union set is significantly smaller than the sum of tokens of all items in $\mathbb{K}_{\boldsymbol{q}}$ (cf. Section 5.2), the proposed token-union-based inference enables highly efficient computation of contextual embeddings. Figure 2 (a) illustrates the difference between the CROSS-JEM encoder, and standard encoders such as monoBERT.

Selective Pooling Layer: Given the contextual embeddings $\mathbf{E}$ for all query tokens $(\mathbb{T}_{\boldsymbol{q}})$ and union over keyword tokens $(\mathbb{T}_{U_{\boldsymbol{q}}})$, CROSS-JEM employs a selective pooling layer to *jointly model* the per-item relevance score. Given $\boldsymbol{k}_j \in \mathbb{K}_{\boldsymbol{q}}$, a pooled representation for pair $(\boldsymbol{q}, \boldsymbol{k}_j)$ is computed as the mean of the contextual embeddings for all tokens in $\mathbb{T}_{\boldsymbol{q}}$ and those tokens in $\mathbb{T}_{U_{\boldsymbol{q}}}$ which are present in $\mathbb{T}_{\boldsymbol{k}_j}$. The set of selected tokens for the pooling layer is given by $\mathbb{P}_{\boldsymbol{q}\boldsymbol{k}_j} = \mathbb{T}_{\boldsymbol{q}} \bigcup \left\{ \mathbf{t}^{[SEP]} \right\} \bigcup \left\{ \mathbb{T}_{U_{\boldsymbol{q}}} \bigcap \mathbb{T}_{\boldsymbol{k}_j} \right\}$.

5

Classifier: The final stage in CROSS-JEM is a d-dimensional shared linear classifier $\boldsymbol{w} \in \mathbb{R}^d$ which computes the relevance score associated with each pair $(\boldsymbol{q}, \boldsymbol{k}_j)$. The selectively pooled representations $\boldsymbol{e}^{\boldsymbol{q}\boldsymbol{k}_j}$ obtained for all $\boldsymbol{k}_j \in \mathbb{K}_{\boldsymbol{q}}$ are batched together ($\boldsymbol{e}^{\boldsymbol{q}\boldsymbol{k}} \in \mathbb{R}^{N \times d}$) allowing for the computation of all logits $[\boldsymbol{f}_{\boldsymbol{q}}]_j = \langle \boldsymbol{w}, \boldsymbol{e}^{\boldsymbol{q}\boldsymbol{k}_j} \rangle$ in a single shot. The scores are defined over these logits (cf. Section 4.1).

4 The CROSS-JEM Algorithm

The detailed inference algorithm for CROSS-JEM is provided in Appendix B, we now describe the Ranking Probability Loss (RPL), which is used to train CROSS-JEM. RPL is a novel variant of the list-based loss function, designed to take advantage of CROSS-JEM's, wherein all item scores are available jointly from a single forward pass.

4.1 The CROSS-JEM Ranking Probability Loss

The standard binary cross-entropy loss ($\mathcal{L}^{BCE}$), used to train cross-encoders, defined over $(\boldsymbol{q}_i, \boldsymbol{k}_j)$, is $-\sum_{i=1}^{|\mathbb{Q}_{tr}|} \sum_{j=1}^{N} \left[y_{ij} \log\left(\boldsymbol{f}_{\boldsymbol{w}, \boldsymbol{q}, i, j}\right) + (1 - y_{ij}) \log\left(1 - \boldsymbol{f}_{\boldsymbol{w}, \boldsymbol{q}, i, j}\right) \right]$, where $\boldsymbol{f}_{\boldsymbol{w}, \boldsymbol{q}, i, j} = \langle \boldsymbol{w}, \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{q}_i, \boldsymbol{k}_j) \rangle$ is the score of item j, associated with query i, computed by means of an inner product with the classifier $\boldsymbol{w}$. However, such cross-entropy-based pointwise losses fail to account for the list of items available for ranking. *List-based loss functions* [21], in contrast, leverage the task-specific ranking information, help learn a scoring function for a list of items to be ranked, rather than for individual query-item pairs. As an example, consider the ListNet [21] loss is given by:

$$\mathcal{L}^{LN}(\boldsymbol{\theta}, \boldsymbol{w}) = -\sum_{i=1}^{|\mathbb{Q}_{tr}|} \sum_{j=1}^{N} P_{\boldsymbol{y}, j} \log\left(P_{\boldsymbol{f}, j}\right), \text{ where } P_{\boldsymbol{x}, j} = \frac{\Phi\left([\boldsymbol{x}]_j\right)}{\sum_{\ell=1}^{N} \Phi\left([\boldsymbol{x}]_\ell\right)}, \tag{1}$$

and $\boldsymbol{x}$ is set either to the targets $[\boldsymbol{y}_{\boldsymbol{q}_i}]$, or the output logits $[\boldsymbol{f}_{\boldsymbol{q}_i}]$, and Φ is a normalizing function, typically the exponential operation, leading to P being a SoftMax function. However, this formulation is still centered around obtaining the pointwise logits, and subsequently computing the top-one probability $P_{\boldsymbol{x}, j}$ using a normalization term that accounts for all pairs.

In CROSS-JEM, we design Ranking Probability Loss, a novel variant of $\mathcal{L}^{LN}$ that factors in all logits $[\boldsymbol{f}_{\boldsymbol{q}_i}]$ computed by taking into account the item-item interactions. Given $\boldsymbol{f}_{\boldsymbol{q}_i}$, the ground-truth scores $\boldsymbol{y}_i$, and a candidate item $\boldsymbol{k}_j$, consider the set $\mathbb{L}_j = \{k \in \{1, N\} : [\boldsymbol{y}_i]_k < [\boldsymbol{y}_i]_j\}$, *i.e.*, $\mathbb{L}_j$ comprises indices k for which the ground-truth score at k is **lower** than $[\boldsymbol{y}_i]_j$, the ground-truth score at j. Then,

$$\mathcal{L}^{RPL} = -\sum_{i=1}^{|\mathbb{Q}_{tr}|} \sum_{j=1}^{N} \left(\sum_{k \in \mathbb{L}_j} [\boldsymbol{y}_i]_k \right) \log\left(\text{SoftMax}\left(\sum_{k \in \mathbb{L}_j} [\boldsymbol{f}_{\boldsymbol{q}_i}]_k \right) \right). \tag{2}$$

The loss $\mathcal{L}^{RPL}$ represents a cross-entropy loss over functions of the target $\boldsymbol{y}_i$ and scores computed as a function of the logits $\boldsymbol{f}_{\boldsymbol{q}_j}$. The following Lemma relates RPL to the ranking probability distribution.

Lemma 1. *(**Ranking Probability Loss**) Assume without loss of generality that $\boldsymbol{f}_{\boldsymbol{q}_i} \in [0, 1]^N$. Let $\mathbf{P} \in \mathbb{R}^{N \times N}$ denote a matrix with entries p_{jk} given by $p_{jk} = \text{Prob}\left(\text{ranking item } \boldsymbol{k}_j \text{ at location } k\right) \triangleq C \sum_{\ell \in \mathbb{L}_k} [\boldsymbol{f}_{\boldsymbol{q}_i}]_\ell$, where C is a normalizing constant. Then, the Ranking Probability Loss maximizes the probability of ranking items $\mathbb{K}_{\boldsymbol{q}_i}$ in the ordering of the ground-truth $\boldsymbol{y}_i$.*

The proof is given in Appendix A. The following Corollary presents an equivalence between the ListNet loss [21] and the proposed Ranking Probability Loss.

Corollary 2. *(**RPL and the ListNet loss**) Minimizing the Ranking Probability Loss is equivalent to optimizing for the ListNet top-1 probability loss (Equation 1 [21]) defined over modified scores $\sum_{k \in \mathbb{L}_j} [\boldsymbol{f}_{\boldsymbol{q}_i}]_j$ and modified ground-truth scores $\tilde{y}_{ij} = \sum_{k \in \mathbb{L}_j} [\boldsymbol{y}_i]_j$.*

Intuitively, defining the modified scores in terms of the sum of all logits in $\mathbb{L}_j$, the set of indices of ground-truth scores lower than the logit at j, ensures that the loss takes into account the interactions between the contextual embeddings contribution to the different logits. This is unique to the CROSS-JEM setting, and Corollary 2 shows that all guarantees derived for ListNet loss also hold for RPL.

We now show that CROSS-JEM, trained with RPL yields significantly better ranking accuracy than existing pointwise and listwise loss functions, leading to state-of-the-art performance.

5 Experimental Validation

Datasets: We evaluate CROSS-JEM on the publicly available on **Stack Overflow Duplicate Questions** [41] **(SODQ)** and a short-text version of **MS MARCO** [42] datasets. While the SODQ dataset is used as is, for MS MARCO, a 'short-text' variant of the query-webpage click dataset is constructed by exclusively considering webpage titles (subsequently referred to as **MS MARCO-Titles**). This narrows the dataset's length distribution, aligning it more closely with the typical item lengths seen in sponsored search. We note that the standard metrics for MS MARCO passage ranking do not apply to MS MARCO-Titles, as they rely on the passage content as well as the title for ranking. Therefore, we report updated numbers for MS MARCO-Titles in Table 2, as appropriate. See Appendix C for more details on the datasets.

Evaluation Metrics: To validate the efficacy of CROSS-JEM for ranking tasks, we consider Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) as the evaluation metrics on both MS MARCO and SODQ datasets (see Appendix D for metric definitions). Note that MAP is a more comprehensive version of MRR, which is specifically designed for scenarios where the test set contains only one positive item per test query. Hence, on MS MARCO, where there is only one relevant item per query, MAP@K and MRR@K are equivalent for any K.

Baselines: We compare CROSS-JEM with transformer based ranking approaches as well as sparse BoW methods such as **BM25**. Within transformer based ranking baselines, we consider state-of-the-art encoder based methods such as **monoBERT** [2]. We also compare CROSS-JEM against more efficient encoders such as **ColBERT** [19]. Dual Encoders such as **ANCE** [35], **DPR** [36], and **INSTRUCTOR** [37] use contrastive learning to train the encoder and obtain dense representations of queries and items. Following Su et al. [37], we use pre-trained and instruction-tuned INSTRUCTOR model for zero-shot evaluation on MS MARCO and SODQ for baseline. We also compare against state-of-the-art sequence-to-sequence class of models containing encoder-decoder architecture, **RankT5** [14]-24 layers. For a fair comparison with 6 layer CROSS-JEM model, we also fine-tune our own RankT5 model with 6 layers (encoder+decoder) on both MS MARCO-Titles and SODQ datasets. For completeness, we also compare against a decoder only ranking model with Llama2-7B architecture, referred to as **RankingGPT-Llama2-7B**, available from Zhang et al. [15].

Hyper-parameters: CROSS-JEM's tunable hyper-parameters include maximum item union length, L_u and number of items per query, N. These hyper-parameter values are dictated by the application requirements and efficiency constraints. We use $N = 10$ and $L_u = 360$ on MS MARCO and $N = 30$ and $L_u = 242$ on SODQ. Hyperparameters used for the baselines are given in Appendix E.

5.1 Accuracy Results on Public Benchmarks

Table 2 shows that CROSS-JEM achieves superior performance over the encoder based methods on both datasets, demonstrating the effectiveness of its listwise ranking approach. Encoder-decoder based methods such as RankT5-6L fine-tuned with a listwise ranking loss, outperform similar sized encoder based methods (monoBERT and ColBERT), highlighting the benefits of listwise modeling for ranking. However, we find that RankT5-base and RankingGPT-llama2-7b, large Seq2Seq models fine-tuned on a long-text passage ranking task [14], perform worse than smaller models that are specifically fine-tuned on short-text ranking. These results suggest that the Seq2Seq models are sensitive to the domain and length of the ranking tasks, and require careful fine-tuning and adaptation. To verify this, we fine-tune a RankT5-base model on the short-text ranking datasets and observe a significant improvement in performance (cf. Appendix E). We also report that CROSS-JEM, which uses a 6-layer BERT as the base encoder, has the same number of parameters as monoBERT, but is much faster and more accurate. Specifically, CROSS-JEM can perform joint (listwise) inference for ranking over $4\times$ faster than monoBERT, which requires multiple pointwise computations (cf. Section 5.2). We perform further experiments on the comparison of pointwise and listwise loss functions in CROSS-JEM in Appendix F.

5.2 Interpreting CROSS-JEM's Performance

We characterize the **time complexity** of CROSS-JEM in terms of number of query tokens $L_q = |\mathbb{T}_q|$, number of item tokens $L_k = |\mathbb{T}_k|$, number of transformer layers L, number of candidate items N, and item union compression factor C (approximated as $\frac{L_k \times N}{|\mathbb{T}_{U_q}|}$). The time complexity for scoring

Table 2: Performance of CROSS-JEM and the baseline methods on the SODQ and MS MARCO-Titles ranking datasets: All baselines and our method, CROSS-JEM, use ∼60-100M parameter base models and are fine-tuned on the corresponding datasets, except for the large pre-trained models (indicated with asterisk (*)), which are used as is without any further fine-tuning on the two datasets. CROSS-JEM surpasses similar-sized state-of-the-art methods fine-tuned for short-text ranking as well as large pre-trained models by at least 3%.

	Method	Parameters	SODQ		MS MARCO-Titles	
			MAP@5	MAP@10	MRR@5	MRR@10
Sparse	BM25	–	32.80	39.26	23.71	24.57
Encoder	Early-interaction monoBERT	66M	46.79	48.04	30.89	32.47
	Late-interaction ColBERT	109M	36.10	37.68	30.25	32.00
	Dual Encoders DPR	66M	47.32	48.48	28.78	30.87
	ANCE	66M	48.31	49.41	28.48	30.53
	INSTRUCTOR*	335M	49.47	50.81	28.84	30.55
Encoder-Decoder	Seq2Seq RankT5-6L	74M	49.50	50.75	30.73	32.52
	RankT5-base*	223M	45.66	49.47	27.87	29.75
Decoder	Ranking LLMs RankingGPT-Llama2-7B*	7B	47.64	50.62	28.66	30.47
Ours	**Joint Ranking CROSS-JEM-6L**	**66M**	**52.40**	**53.05**	**33.82**	**35.45**

Table 3: A comparison of latency between CROSS-JEM and various baselines. The mean latency for scoring 700 items per query was computed on A100 GPUs for all models. We observe that CROSS-JEM takes about $4\times$ lower inference time than standard cross-encoders (monoBERT).

Method	ANCE	ColBERT	monoBERT	RankT5-6L	**CROSS-JEM**
Latency (ms) ↓	4.0	4.5	41.3	41.3	9.8

all N items jointly is $(L_{\boldsymbol{q}} + L_{\boldsymbol{k}} N/C)^2 L$. On the other hand, for the standard cross-encoder, the corresponding time complexity is $(L_{\boldsymbol{q}} + L_{\boldsymbol{k}})^2 LN$. In practice, the inference time depends on factors such as the implementation of the model, the hardware employed, and optimizations used (such as quantization). For the sponsored search setting detailed in Appendix G, assuming $L_{\boldsymbol{q}} \approx L_{\boldsymbol{k}}$, letting $N = 100$, the maximum tokens in an item is 24 and maximum item union length used is 220, we have $C \approx (24 * 100)/220 \approx 11$. Then, the standard cross-encoder time complexity is $400 L_q^2 L$, while that of CROSS-JEM is $101.8 L_q^2 L$, which is $3.9\times$ lower. These inference time gains are also reflected in the latency comparison of CROSS-JEM against pointwise approaches, *e.g.,* monoBERT (cf. Table 3).

6   Conclusion: Limitations and Future Work

We introduced CROSS-JEM, an accurate and efficient approach for joint ranking of a set of short-text items for a given query based on relevance. CROSS-JEM effectively addresses the two major challenge in existing ranking architectures – sub-optimal accuracy due to pointwise inference, and significantly higher computational cost. Our extensive evaluations on publicly available ranking benchmarks as well as large-scale sponsored search datasets reveal that CROSS-JEM significantly outperforms the baselines, establishing a new state of the art. The scope of this work is primarily focused on the ranking of short texts, a common requirement in both industrial Sponsored Search applications and academic benchmarks, including tasks like matching queries with webpage titles and ranking duplicate questions. While the current work demonstrates significant gains on such short-text ranking tasks, the proposed approach could be adapted for long-text ranking by incorporating post-hoc positional encodings (cf. Appendix F) and more sophisticated attention mechanisms that account for longer document lengths. Exploring these adaptations is an area for future research. CROSS-JEM opens up new directions for designing accurate ranking architectures and algorithms, accounting for application-specific constraints.

References

[1] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. Cascade ranking for operational e-commerce search. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1557–1565, 2017.

[2] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert, 2020.

[3] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense text retrieval based on pretrained language models: A survey. *ACM Trans. Inf. Syst.*, 42(4), feb 2024. ISSN 1046-8188. doi: 10.1145/3637870. URL https://doi.org/10.1145/3637870.

[4] J. Lin, R. Nogueira, and Yates A. Pretrained Transformers for Text Ranking: BERT and Beyond. In *NAACL*, 2021.

[5] Yucheng Zhou, Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Guodong Long, Binxing Jiao, and Daxin Jiang. Towards robust ranker for text retrieval. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5387–5401, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.332. URL https://aclanthology.org/2023.findings-acl.332.

[6] Yixing Fan, Xiaohui Xie, Yinqiong Cai, Jia Chen, Xinyu Ma, Xiangsheng Li, Ruqing Zhang, Jiafeng Guo, et al. Pre-training methods in information retrieval. *Foundations and Trends® in Information Retrieval*, 16(3):178–317, 2022.

[7] Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. Semantic models for the first-stage retrieval: A comprehensive review. *ACM Transactions on Information Systems (TOIS)*, 40(4):1–42, 2022.

[8] Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. High accuracy retrieval with multiple nested ranker. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 437–444, 2006.

[9] K. Dahiya, D. Saini, A. Mittal, A. Shaw, K. Dave, A. Soni, H. Jain, S. Agarwal, and M. Varma. DeepXML: A Deep Extreme Multi-Label Learning Framework Applied to Short Text Documents. In *WSDM*, 2021.

[10] H. Zhou, M. Huang, Y. Mao, C. Zhu, P. Shu, and X. Zhu. Domain-constrained advertising keyword generation. In *WWW*, 2019.

[11] Peter Schonhofen. Identifying document topics using the wikipedia category network. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, pages 456–462, 2006. doi: 10.1109/WI.2006.92.

[12] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*, 2019.

[13] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.63. URL https://aclanthology.org/2020.findings-emnlp.63.

[14] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 2308–2313, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394086. doi: 10.1145/3539618.3592047. URL https://doi.org/10.1145/3539618.3592047.

[15] Longhui Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, Meishan Zhang, and Min Zhang. A two-stage adaptation of large language models for text ranking. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics ACL 2024*, pages 11880–11891, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.706.

[16] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.97. URL https://aclanthology.org/2024.findings-naacl.97.

[17] Junyan Chen, Frédéric Dubut, Jason (Zengzhong) Li, and Rangan Majumder. Make every feature binary: A 135b parameter sparse neural network for massively improved search relevance, 2023. URL https://tinyurl.com/y9amxjau.

[18] Wenhao Lu, Jian Jiao, and Ruofei Zhang. Twinbert: Distilling knowledge to twin-structured compressed bert models for large-scale retrieval. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 2645–2652, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3412747. URL https://doi.org/10.1145/3340531.3412747.

[19] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

[20] Luyu Gao, Zhuyun Dai, and Jamie Callan. Rethink training of bert rerankers in multi-stage retrieval pipeline. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*, page 280–286, Berlin, Heidelberg, 2021. Springer-Verlag. ISBN 978-3-030-72239-5. doi: 10.1007/978-3-030-72240-1_26. URL https://doi.org/10.1007/978-3-030-72240-1_26.

[21] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273513. URL http://doi.acm.org/10.1145/1273496.1273513.

[22] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937, 2023.

[23] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*, 2023.

[24] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, et al. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518, 2024.

[25] Tao Qin, Tie-Yan Liu, and Hang Li. Query-level loss functions for information retrieval. *Information Processing Management*, 44(2):838–855, January 2008. URL https://www.microsoft.com/en-us/research/publication/query-level-loss-functions-for-information-retrieval/.

[26] Wei Chen, Tie-yan Liu, Yanyan Lan, Zhi-ming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper_files/paper/2009/file/2f55707d4193dc27118a0f19a1985716-Paper.pdf.

[27] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. Cross-domain modeling of sentence-level evidence for document retrieval. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3490–3496, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1352. URL https://aclanthology.org/D19-1352.

[28] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2421–2425, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657951. URL https://doi.org/10.1145/3626772.3657951.

[29] Luyu Gao, Zhuyun Dai, and Jamie Callan. Rethink training of bert rerankers in multi-stage retrieval pipeline. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part II 43*, pages 280–286. Springer, 2021.

[30] Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. Zero-shot listwise document reranking with a large language model. *arXiv preprint arXiv:2305.02156*, 2023.

[31] Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 38–47, 2024.

[32] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*, 2023.

[33] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2421–2425, 2024.

[34] Omar Khattab, Christopher Potts, and Matei Zaharia. Baleen: Robust multi-hop reasoning at scale via condensed retrieval. *Advances in Neural Information Processing Systems*, 34: 27670–27682, 2021.

[35] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*, 2021.

[36] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-T. Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

[37] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023. findings-acl.71. URL https://aclanthology.org/2023.findings-acl.71.

[38] S. J. Subramanya, Devvrit, R. Kadekodi, R. Krishnaswamy, and H. Simhadri. DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node. In *NeurIPS*, 2019.

[39] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[40] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, pages 3715–3734, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.272. URL https://aclanthology.org/2022.naacl-main.272.

[41] Xueqing Liu, Chi Wang, Yue Leng, and ChengXiang Zhai. Linkso: a dataset for learning to retrieve similar question answer pairs on software development forums. In *4th ACM SIGSOFT International Workshop on NLP for Software Engineering*, pages 2–5, 11 2018. doi: 10.1145/3283812.3283815.

[42] Zhuyun Dai and Jamie Callan. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*, pages 1897–1907, 2020.

[43] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[45] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022. doi: 10.48550/ARXIV.2210.07316. URL https://arxiv.org/abs/2210.07316.

[46] Andrew Trotman, Antti Puurula, and Blake Burgess. Improvements to bm25 and language models examined. In *Proceedings of the 19th Australasian Document Computing Symposium*, ADCS '14, page 58–65, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450330008. doi: 10.1145/2682862.2682863. URL https://doi.org/10.1145/2682862.2682863.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A   Proofs of Lemma 1 and Corollary 2

To derive the proof, we recall both Lemma 1 and the Ranking Probability Loss:

$$\mathcal{L}^{RPL} = - \sum_{i=1}^{|\mathbb{Q}_{tr}|} \sum_{j=1}^{N} \left(\sum_{k \in \mathbb{L}_j} [\boldsymbol{y}_i]_k \right) \log \left(\text{SoftMax} \left(\sum_{k \in \mathbb{L}_j} [\boldsymbol{f}_{\boldsymbol{q}_i}]_k \right) \right),$$

where $\tilde{s}_{ij} = \sum_{k \in \mathbb{L}_j} [\boldsymbol{f}_{\boldsymbol{q}_i}]_j$ is the modified score associated with the query-item pair $(\boldsymbol{q}_i, \boldsymbol{k}_j)$. The following Lemma sheds light on the relationship between RPL and ranking probability distribution.

Lemma. *(**Ranking Probability Loss**) Let $\mathbf{P} \in \mathbb{R}^{N \times N}$ denote a matrix with entries p_{jk} given by $p_{jk} = \text{Prob}\left(\text{ranking item } \boldsymbol{k}_j \text{ at location } k \right) \triangleq C \sum_{\ell \in \mathbb{L}_k} [\boldsymbol{f}_{\boldsymbol{q}_i}]_\ell$, where C is a normalizing constant. Then, the Ranking Probability Loss maximizes the probability of ranking queries $\mathbb{K}_{\boldsymbol{q}_i}$ in the ordering of the ground-truth ranking $\boldsymbol{y}_i$.*

Proof. Given $\boldsymbol{f}_{\boldsymbol{q}_i}$, the ground-truth scores $\boldsymbol{y}_i$, and a candidate item $\boldsymbol{k}_j$, we define the set $\mathbb{L}_j = \{k \in \{1, N\} : [\boldsymbol{y}_i]_k < [\boldsymbol{y}_i]_j\}$, *i.e.*, $\mathbb{L}_j$ comprises the indices k for which the ground truth score of $[\boldsymbol{y}_i]_j$ is larger than the ground truth score at location k. Additionally, without loss of generality, we assume that $\boldsymbol{f}_{\boldsymbol{q}_i}$ and $\boldsymbol{y}_i$ are normalized to have entries that lie in $[0, 1]$. We have:

$$p_{jk} = \text{Prob}\left(\text{ranking item } \boldsymbol{k}_j \text{ at location } k\right) \triangleq C \sum_{\ell \in \mathbb{L}_k} [\boldsymbol{f}_{\boldsymbol{q}_i}]_\ell,$$

Where C is an appropriately chosed constant, such that $\mathbf{P}$ is a matrix with entries summing to one. Further, in the context of ranking with the logits/scores, we have:

$$p_{jk} = \text{Prob}\left(\text{ranking item } \boldsymbol{k}_j \text{ at location } k\right)$$
$$= \text{Prob}\left(\text{logit of item } \boldsymbol{k}_j > \{\text{logits of all items } \boldsymbol{k}_k \text{ such that } k \in \mathbb{L}_k \}\right).$$

Since the entire analysis is presented in the context of a each query $\boldsymbol{q}_i$, for convenience, we abuse notation, and denote $[\boldsymbol{f}_{\boldsymbol{q}_i}]_j = [\boldsymbol{f}]_j = f_j$. Then, we have

$$p_{jk} = \text{Prob}\left(\text{logit of item } \boldsymbol{k}_j > \{\text{logits of all items } \boldsymbol{k}_k \text{ such that } k \in \mathbb{L}_k \}\right)$$
$$= \text{Prob}\left(f_j > \{f_k \text{ for all } k \in \mathbb{L}_k\}\right)$$
$$= \text{Prob}\left(f_j > \max_{k \in \mathbb{S}_k}\{f_k\}\right)$$
$$\approx \text{Prob}\left(f_j > \sum_{\ell \in \mathbb{L}_k} f_\ell\right), \tag{3}$$

where the last step is a consequence of the norm-bound $\|\boldsymbol{f}\|_1 \leq \|\boldsymbol{f}\|_\infty$. First, we note that, given the condition in Equation 3, for all k ranked higher than j, $p_{jk} = 0$. Further, for all k ranked lower than j, it suffices to show that the probability p_{jk} is non-zero, for $j = k$. To verify this, consider two probabilities p_{ik_1} and p_{ik_2}, with k_1 ranked higher than k_2. Let $\mathbf{1}_{ik_1}$ denote the indicator of the event associated with p_{ik_1}. We have

$$p_{ik_2} \approx \text{Prob}\left(f_j > \sum_{\ell \in \mathbb{L}_{k_2}} f_\ell \ \Big|\ \mathbf{1}_{ik_1}\right) \text{Prob}(\mathbf{1}_{ik_1}) = p_{ik_1}$$

To build intuition for the this, assume without loss of generality that $\boldsymbol{f}$ have been sorted in a non-increasing manner. Let $j = 1$ $k_1 = 1$ and $k_2 = 3$. Then, it is clear to see that

$$\text{Prob}\left(f_1 > f_4 + f_5 + \ldots \ \Big|\ f_1 > f_2 + f_3 + f_4 + f_5 + \ldots\right) \text{Prob}\left(f_1 > f_2 + f_3 + \ldots\right)$$
$$= \text{Prob}\left(f_1 > f_2 + f_3 + \ldots\right).$$

Given $\mathbf{P}$ built as described above, and $\mathbf{P}^*$ defined similarly over scores $\boldsymbol{y}$, from the discussion above, we see that minimizing the distance between $\mathbf{P}$ and $\mathbf{P}^*$ is equivalent to minimizing the distance between the vectors $\boldsymbol{p} = \text{Diag}\left(\mathbf{P}\right)$ and $\boldsymbol{p}^* = \text{Diag}\left(\mathbf{P}^*\right)$. We observe that, when normalized, the entries of $\boldsymbol{p}$ correspond to the probability of ranking item $\boldsymbol{k}_j$ and rank j, given sorted vectors $\tilde{\boldsymbol{f}}$ and $\tilde{\boldsymbol{y}}$. Then, the RPL is the binary cross entropy loss defined between $\boldsymbol{p}$ and $\boldsymbol{p}^*$, and represents minimizing the KL-divergence between the predicted ranking probability distribution $\tilde{\boldsymbol{f}}$ and the ground-truth ranking distribution $\tilde{\boldsymbol{y}}$, up to a normalizing constant factor. This completes the proof of Lemma 1. $\square$

Proof of Corollary 2: To link the Ranking Probability Loss to the ListNet loss [21], we recall that:

$$\mathcal{L}^{LN} = -\sum_{i=1}^{|\mathbb{Q}_{tr}|} \sum_{j=1}^{N} P_{\boldsymbol{y},j} \log\left(P_{\boldsymbol{s},j}\right),$$

where $P_{\boldsymbol{x},j} = \dfrac{\Phi\left([\boldsymbol{x}]_j\right)}{\sum_{\ell=1}^{N} \Phi\left([\boldsymbol{x}]_\ell\right)}$ Setting $[\tilde{\boldsymbol{s}}_{\boldsymbol{q}_i}]_j = \tilde{s}_{ij} = \sum_{\ell \in \mathbb{L}_j} [\boldsymbol{f}_{\boldsymbol{q}_i}]_\ell$, and $[\tilde{\boldsymbol{y}}_i]_j = \tilde{y}_{ij} = \sum_{\ell \in \mathbb{L}_j} [\boldsymbol{y}_i]_\ell$ and a Φ that result in mapping $\tilde{\boldsymbol{s}}$ and $\tilde{\boldsymbol{y}}$ to be valid probabilities distributions, we see the equivalence between the ListNet loss, and the proposed Ranking Probability Loss.

B The CROSS-JEM Inference

The procedure for forward pass of a CROSS-JEM model during training as well as inference is outlined in Algorithm 2, while the procedure to obtain the attention map over the item union set $\mathbb{T}$ is described in Algorithm 1.

Algorithm 1 Method to create attention masks for item k_i. **Input**: T_q: Tokenized Query, T_U: Tokens in item Union Set, k: tokenized k_i keyword. **Output**: AttMask: Attention Mask for the keyword

---

1: **procedure** GETKUATTENTIONMASK(T_q, T_U, k)
2:     AttMask $\leftarrow []$
3:     **for** i from 0 to LEN(T_q) - 1 **do**
4:         AttMask.ADDITEM(1)
5:     **end for**
6:     **for** i from 0 to LEN(T_U) - 1 **do**
7:         **if** $T_U[i]$ in k **then**
8:             AttMask.ADDITEM(1)
9:         **else**
10:        AttMask.ADDITEM(0)
11:        **end if**
12:     **end for**
13:     **return** AttMask
14: **end procedure**

C Datasets

MS MARCO: MS MARCO document re-ranking dataset contains queries and their clicked web pages consisting of webpage title, URL, and passage. We create a śhort-textv́ersion of the dataset by considering only the titles of the webpages, making the length statistics of the dataset better aligned with the real-world applications of ranking in sponsored search. We experiment with HDCT [42] retriever based training dataset consisting of 0.37M training queries and top 10 predictions from HDCT along with their ground truth click labels. We use the dev set to report our metrics as this set has ground truth labels available for evaluation. We use $\sim$3.7M training pairs available in MS MARCO HDCT dataset sourced from [42] as described above for training CROSS-JEM and baselines using DistilBERT [43]. The target scores for all training pairs are obtained from a monoBERT model trained on binary ground truth click data with BERT-Base [44] as the base encoder for MS MARCO dataset.

SODQ: Stack Overflow Duplicate Questions dataset involves ranking questions on Stack Overflow as duplicates or not with the tags Java, JavaScript and Python [41]. It is also one of the re-ranking datasets on the popular MTEB benchmark [45]. StackOverflowDupQuestions on MTEB benchmark is the only shoft text re-ranking dataset with both training and evaluation data available, and is hence used in our experiments. Similar to our experiments on MS MARCO dataset, the target scores for training CROSS-JEM as well as baselines are obtained from a BERT [44] based monoBERT model trained on binary relevance of duplicate questions.

Sponsored Search Dataset: The training dataset for sponsored search query to advertiser matching task is created using a BERT-Large based monoBERT model trained on manually labeled and good-click data as the teacher model. A query-item (advertiser bid keyword) pair in the good click data is

Algorithm 2 Getting relevance scores for a query and retrieved set of items using CROSS-JEM.
Input: Query q, retrieved set of N items for q where $K_q = \{k_0, k_1, ..., k_{N-1}\}$. **Output**: Scores
$S = \{s_0, s_1, ..., s_{N-1}\}$

1: $T_q \leftarrow$ TOKENIZE(Q)   ▷ Tokenize the query
2: $T_U \leftarrow \{\}$
3: $K_{tokens} \leftarrow []$ ▷ Store tokenized items
4: **for** k in K_q **do**
5:     $k_{tokens} \leftarrow$ TOKENIZE(k)
6:     $T_U \leftarrow$ UNION(T_U, k_{tokens})
7:     K_{tokens}.ADDITEM(k_{tokens})
8: **end for**
9: $T_U \leftarrow$ SORTED(T_U)
10: KUAttMask $\leftarrow []$ ▷ KUAttMask: item Union Attention Mask
11: **for** i from 0 to $N-1$ **do**
12:     AttMask $\leftarrow$ GETKUATTENTIONMASK($T_q, T_U, K_{tokens}[i]$) (cf. Algorithm 1)
13:     KUAttMask.ADDITEM(AttMask)
14: **end for**
15: sepToken $\leftarrow$ TOKENIZE($[SEP]$)   ▷ Token id for [SEP] token
16: encInpToks $\leftarrow T_Q$ ▷ Tokens to be passed through the Encoder
17: encInpToks.ADDITEM(sepToken)
18: **for** t_U in T_U **do**
19:     encInpToks.ADDITEM(t_U)
20: **end for**
21: $E \leftarrow$ ENCODER(encInpToks)
22: $S \leftarrow$ SELECTIVEPOOLING(E, KUAttMask)   ▷ $S \in \mathbb{R}^{N \times d}$
23: $S \leftarrow$ CLASSIFIER(S)   ▷ $S \in \mathbb{R}^N$
24: **return** S

---

obtained when the user clicked on the ad corresponding to an advertiser keyword in response to their query, and did not close the ad quickly indicating they found it relevant. This BERT-Large teacher model was used to score 100 predicted items each for 18.6M queries on the search engine during a time period. This resulted in around 1.8B query-item pairs with scores in 0 to 1 range as training data for CROSS-JEM and all baselines in Table 7.

D Metrics

- **Mean Average Precision (MAP)**: This is a ranking metric defined as the mean of Average Precision (AP) over the positive and negative detected classes:

$$\frac{1}{|Q|} \sum_{u=1}^{|U|} \left(\frac{1}{m} \sum_{k=1}^{N} P_u(k) \cdot rel_u(k) \right) \tag{4}$$

where $|Q|$ is the total number of queries, $P_u(k)$ is the precision at cut-off k in the list, $rel_u(k)$ is an indicator function equaling 1 if the item at rank k is a relevant document, otherwise zero.

- **Mean Reciprocal Rank (MRR)**: Rank is defined as the position of the first relevant item in the ranked list. MRR is hence defined as below:

$$\frac{1}{|Q|} \sum_{i=1}^{|Q|} \left(\frac{1}{rank_i} \right) \tag{5}$$

- **Accuracy**: Positive, Negative, and Overall Accuracy denote the proportion of positive, negative, and overall instances, respectively, in the test set that are accurately identified.

- **Area Under the ROC Curve (AUC-ROC)**: The ROC curve is a plot of True Positive Rate (TPR) or sensitivity against False Positive Rate (FPR) at different thresholds.

Table 4: Improvement in performance of a large Seq2Seq model, RankT5-base after fine-tuning on short-text ranking benchmarks

Method	SODQ		MS MARCO	
	MAP@5	MAP@10	MRR@5	MRR@10
RankT5-base (pre-trained)	45.66	49.47	27.87	29.75
RankT5-base (fine-tuned)	55.9	56.8	33.72	35.14

Table 5: Hyperparameters used in CROSS-JEM.

Hyperparam	SODQ	MS MARCO	Sponsored Search
N	30	10	100
L_u	265	360	262

Table 6: (a) Ablation on loss function in CROSS-JEM; (b) Adding item token sequence information in CROSS-JEM via positional encodings in the pooling layer.

Method	BCE	CE	ListNet	RPL
MRR @10	31.46	32.03	30.27	35.45

(a)

Method	Without Positional Encodings	With Positional Encodings
MRR@10	35.45	35.68
MRR@5	33.82	33.98

(b)

E   Baselines and Hyperparameters

For the monoBERT, DPR, and ANCE baselines, we tune the following hyperparameters based on the validation set accuracy: (a) learning rate; (b) weight decay, and (c) number of epochs. ColBERT is trained and evaluated with the default set of hyperparameters provided by Khattab and Zaharia [19][4]. We use the pre-trained checkpoint[5] and code-base[6] provided by Su et al. [37] for the INSTRUCTOR model, and test its zero-shot performance using the instructions mentioned in the paper. CROSS-JEM is trained with exactly same setting as monoBERT: learning rate of 1e-4, linear learning rate scheduler, and AdamW optimizer. Hyperparameters specific to CROSS-JEM (N and L_u) are provided in Table 5. The metrics for BM25 on SODQ are taken from [41], while they are computed following the procedure presented by Trotman et al. [46] on the MS MARCO dataset.

Fine-tuning RankT5-base (24L) for Short-text Ranking: We fine-tune the model checkpoint available from Zhang et al. [15] on short-text ranking benchmarks and note the performance improvements in Table 4.

E.1   Compute

All baselines on MS MARCO and SODQ datasets including CROSS-JEM were trained on 8 V100 GPUs. Experiments on proprietary Sponsored Search dataset were conducted on larger GPU cluster with 16 V100s.

F   Ablation Experiments on Public Datasets

Loss Function: We demonstrate performance comparison of pointwise and listwise loss functions with CROSS-JEM architecture in Table 6 (a). While listwise loss functions (Cross-Entropy and ListNet [21]) either perform similar to or slightly better than pointwise losses (such as Binary Cross-Entropy), CROSS-JEM trained with RPL performs better than pointwise or listwise losses.

Incorporating Token Sequence Order Information: CROSS-JEM encodes the union of tokens from all ranking items, which enables fast inference by reducing the input sequence length. However, this also discards the original token order information within each item, treating them as bags of tokens. Though in our preliminary experiments, we observe that the loss of sequence information in the encoder has a negligible impact on the accuracy particularly in short-text ranking tasks (within 1%, refer Appendix G). However, token sequence information could be crucial in many ranking scenarios, and could get completely ignored in CROSS-JEM. To address this limitation, we propose an extension of CROSS-JEM that leverages sinusoidal positional encodings [47] to inject the item sequence information back into the model via the selective pooling layer. The key idea is to preserve the original position indices of the tokens for each item in the ranking list, remove them during the encoding process, and then add them back to the corresponding token context vectors after the CROSS-JEM encoder. The positional encodings are computed based on the original position indices and are summed with the token context vectors. The resulting vectors are then pooled together for classification as described in Section 3. This simple yet effective technique improves the MRR@10 by 0.2% on the MS MARCO dataset (cf. Table 6 (b)), without any additional latency. This technique could also be used to improve CROSS-JEM's performance on long texts, which we leave for future work.

G   Experiments: Sponsored Search Dataset

In large-scale search and recommendation systems like sponsored search, the ranking model serves to weeding out bad retrievals and rank the prediction pool of different retrievers to select the top-k. We evaluate the effectiveness of CROSS-JEM on this real-world task of matching user queries to relevant advertiser-bid keywords. A large scale dataset consisting of 1.8B query-keyword pairs was created by mining search engine logs (detailed in Appendix C).

Accuracy comparison: As shown in Table 7 CROSS-JEM improves over the in-production sparse neural model MEB [17] in MAP by over 13%. Further, CROSS-JEM also outperforms ANCE and TwinBERT by large margins in MAP, Precision, and Recall. We also assess CROSS-JEM's ability to eliminate irrelevant items while preserving relevant ones. Table 7 presents the negative and overall accuracy when retaining top 80% of positive items per query. CROSS-JEM achieves 99.45% negative accuracy, removing nearly all irrelevant items.

Efficiency Gains: We observe that CROSS-JEM takes only 9.8 ms to score 700 keywords for a query on a A100 GPU. In contrast, monoBERT takes 41.3 ms for the same task, rendering it unsuitable for online deployment. This represents a *more than 4-fold reduction in latency* compared to standard cross-encoder models. The latency gains are because CROSS-JEM scores multiple items for a query in one shot by passing their concatenated tokens through the model. On the other hand, monoBERT scores each query-item pair independently necessitating 700 passes compared to CROSS-JEM's 7 passes. Additionally, CROSS-JEM provides 3× lower latency on GPUs than MEB on CPUs. This highlights CROSS-JEM's ability to leverage GPU acceleration for efficiency, unlike sparse models.

CROSS-JEM achieves a high throughput of 17,200 query-keyword pairs per second. This is over 5× more than the 3,350 pairs per second for monoBERT. The massive throughput and latency gains show CROSS-JEM's ability to meet the computational demands of large-scale industrial systems without sacrificing accuracy.

Understanding CROSS-JEM Efficiency Gains: To better understand **the efficiency gains** in CROSS-JEM, we analyze the effect of the significant token overlap amongst candidate items in the

---

[4]https://github.com/stanford-futuredata/ColBERT
[5]https://huggingface.co/hkunlp/instructor-base
[6]https://github.com/xlang-ai/instructor-embedding

Table 7: Comparison of CROSS-JEM with production baselines on Sponsored Search Ads Dataset for ranking advertiser-bid keywords for a user query. CROSS-JEM outperforms baseline methods (with a latency small enough to be deployed for real-time ranking) by 13% in MAP. CROSS-JEM filters >90% irrelevant predictions at a threshold which retains 80% of good predictions.

Method	MAP@100	P@50	R@50	AUC	Negative Accuracy	Overall Accuracy
ANCE	78.39	41.94	94.02	89.84	86.19	85.55
TwinBERT	83.56	43.50	95.36	92.10	90.60	88.58
MEB	84.38	42.94	94.65	91.77	84.59	84.40
CROSS-JEM	**97.48**	**45.76**	**99.07**	**99.41**	**99.45**	**95.27**

set $\mathbb{K}_q$ for a given query q. We trained an ANCE [35] dense retriever on the same train set as above. For a query $q \in \mathbb{Q}_{te}$, recall that $\mathbb{K}_q = \{k_1, k_2, \ldots, k_N\}$ are the top N (=100 in our experiments) items retrieved using ANCE. Let $\mathbb{T}_{k_j}^L$ denote word-piece tokens in k_j with a max-length of L. Let $\mathbb{T}_U$ denote the union of all tokens of items $k_j \in \mathbb{K}_{te}$. We compute the following statistics:

$$ m = \frac{1}{|\mathbb{Q}_{te}|} \sum_{q \in \mathbb{Q}_{te}} \left(\sum_{j=1}^{N} |\mathbb{T}_{k_j}^L| \right), \quad \text{and} \quad N_u = \frac{1}{|\mathbb{Q}_{te}|} \sum_{q \in \mathbb{Q}_{te}} \left(\left| \bigcup_{j=1}^{N} \mathbb{T}_{k_j}^L \right| \right), $$

where m is the *mean total tokens* and N_u is the mean size of the set $\mathbb{T}_U$. Intuitively, m is the the sum of item token lengths on average, while N_u is the cardinaltiy of the union set, averaged over the queries q for which the candidate items $\mathbb{K}_q$ were obtained. If the items k_i have significant overlap, $m \gg N_u$. Statistically, N_u is found to be at least $5\times$ smaller than m, indicating high redundancy, and correlates with observations on the *sponsored search data* (cf. Appendix Table 8 (b)).

Online A/B testing: CROSS-JEM was deployed in the ranking stage of a premier search engine to conduct A/B tests on live traffic. The ranking stage receives an average of 700 keywords and up to 1400 keywords in the 99th percentile, from a suite of retrieval algorithms. The control group consisted of a proprietary combination of late interaction, dense retrieval, and sparse-neural-network algorithms. CROSS-JEM demonstrated a decrease in the quick-back-rate (users who close the ad quickly, indicating non-relevance) by over 1.8%. Furthermore, as judged by expert judges, CROSS-JEM improved the proportion of accurate predictions by 10.2%.

We compare CROSS-JEM against methods that can be deployed for real-time ranking including ANCE, MEB, and TwinBERT. TwinBERT is a lighter version of ColBERT. It applies an MLP layer to individual query and keyword embeddings, unlike ColBERT which considers interactions along all token embeddings. This makes TwinBERT more efficient and practical for real-world systems due to lower storage requirements. Table 7 shows the comparison of CROSS-JEM with baseline methods in production where CROSS-JEM outperforms existing methods by large margins.

Ablation on Number of items per Query (N): From Table 9, we vary the number of items scored per query from 10 to 100. With more items, the token overlap increases, providing CROSS-JEM more opportunity for joint modeling. Correspondingly, we observe gains in negative accuracy and AUC as items per query increase.

Ablation on Encoder $\mathcal{E}_\theta$ in CROSS-JEM: Table 10 shows that even with a smaller encoder, CROSS-JEM provides significant accuracy gains over sparse models like MEB while having low latency.

Ablation on Sequence Information: We analyze the effect of the ordering/sequencing of the item text on classification accuracy using a cross-encoder model; and a train dataset consists of about 100M query-item pairs (q, k), drawn from $\mathbb{Q}_{tr} \times \mathbb{I}_{tr}$, mined from proprietary search engine logs. Given (q, k), we compare two cross-encoder $\mathcal{E}$ models: 1) $\mathcal{E}_{CE}$: Standard cross-encoder scoring the pairs (q, k), and 2) $\mathcal{E}'_{CE}$: Cross-encoder trained to score pairs (q, k'), where the item k' is obtained by sorting the tokens in k alphabetically.

The hypothesis is that, if the cross-encoders $\mathcal{E}_{CE}$ and $\mathcal{E}'_{CE}$ have similar scoring accuracy, then the sequence ordering is relatively less informative for this task. While testing $\mathcal{E}'_{CE}$ is evaluated on (q, k') pairs k' is drawn from $\mathbb{I}_{te}$, with its tokens sorted alphabetically. Table 8(a) shows how the variants perform on a test set of 10M pairs. We observe that, when the sequence information in k is

Table 8: Sponsored search dataset statistics that motivate our two key insights. (a) Cross-encoder trained with with ordered tokens ($\mathcal{E}_{CE}$) and alphabetically sorted tokens for all items ($\mathcal{E}'_{CE}$). The small performance delta (between rows) indicates that sequence ordering is not critical. (b) The mean total tokens m in a retrieved item set Q_{t_e} and the mean size N_u of tokens in the union of Q_{t_e}. The ratio of m to N_u being ~ 5 indicates strong token overlap.

Algorithm	MAP@100
$\mathcal{E}_{CE}$	93.03
$\mathcal{E}'_{CE}$	92.76

(a)

Max-length in word-piece (L)	m	N_u
12	498.38	93.70
16	499.34	94.02
32	501.52	94.60

(b)

Table 9: Variation in accuracy on varying the number of items scored per query by CROSS-JEM. We observe only minor variation in changing the number of items to be ranked at inference time. This observation is useful in real-world ranking which receive item candidates from a set of retrieval algorithms and hence the number of items to be scored can vary with the query.

N	Negative Accuracy	Overall Accuracy	AUC
10	99.18	94.94	99.24
20	99.37	94.96	99.34
50	99.52	94.82	99.40
80	99.56	94.71	99.51
100	99.45	95.27	99.42

Table 10: Variation in accuracy with base encoder $\mathcal{E}_{\boldsymbol{\theta}}$ in CROSS-JEM.

Encoder	Negative Accuracy	Overall Accuracy	AUC	Latency CPU
TinyBERT - 2 layer	96.52	92.76	96.67	114.3
DistilBERT - 6 layer	99.45	95.27	99.61	744.7

discarded, both the mean average precision (MAP) and accuracy are within 1% of the case when the sequence information is retained.

H   Qualitative Analysis

We present additional examples comparing the ranking performance of CROSS-JEM and the baseline cross encoder model in Table 11.

Table 11: A comparison of the top-5 ranked items obtained using CROSS-JEM's listwise modeling and a pointwise ranking model [2]. Relatively more *generic* (but still relevant) items are ranked higher in baseline predictions, owing to their frequency in training data, token-level matching and other biases, which are circumvented when using listwise architectures capable of evaluating all the shortlisted items in a single forward pass, and rank the more relevant (and specific) items higher.

Query	Top-5 Ranked Item in the Proposed Approach (CROSS-JEM)	Top-5 Ranked Item in a Baseline Cross Encoder
different foods of oaxaca mexico	'the foods of oaxaca' 'oaxacan cuisine' 'exploring oaxacan food' 'authentic recipes from oaxaca, mexico' '6 things you'll love about oaxaca'	'culinary tales: the kinds of food mexicans eat' 'mexican christmas foods' 'mexican cuisine' 'culture: food and eating customes in mexico' 'popular food in mexico'
what is the bovine growth hormone	'recombinant bovine growth hormone' 'what is rbgh?' 'rbgh' 'bovine growth hormone and milk : what you need to know' 'what is rbst?'	'growth hormone' 'human growth hormone' 'alternative names for growth hormone' 'human growth hormone and insulin are friends' 'growth hormone (somatotropin)'
what is the state nickname of new mexico	'what are the nicknames of the state of new mexico?' 'state nicknames new mexico – south carolina and their explanation' 'what is new mexico's nickname?' 'new mexico state names (etymology of names)' 'the state of new mexico'	meh 'the state of new mexico' meh 'state of new mexico' 'new mexico' 'new mexico state university' 'state of mexico'
is the elliptical bad for your knees	'does an elliptical make bad knees worse?' 'are the elliptical machines bad for your knees?' 'is an elliptical the best machine for knees that are chronically painful?' 'why does my knee hurt on an elliptical machine?' 'elliptical machine is good for osteoarthritis of the knee!'	'what exercises can help relieve knee pain?' '4 bad exercises for bad knees' 'how to treat a knee sprain' 'how to strengthen legs with bad knees' 'yoga bad for your knees, indian doctor warns'
what do you use for oxygen facial machines	'oxygen facials and other skincare services' 'oxygenating facial treatment' 'oxygen facial : home kits, beauty benefits, side effects, process' 'benefits of oxygen facial' '4 beauty - boosting benefits of oxygen facials'	'using oxygen safely' 'the oxygen machine' 'shop cpap and oxygen' 'oxygen concentrators and generators' 'oxygen concentrators & generators'